次世代ものつくりプラットフォーム (HPC/PF)
PDI(パラメータ空間設計・入力支援) サブシステム
操作説明書

Version 1.4.4

2015 年 10 月

東京大学 生産技術研究所

改版履歴

リリース	版数	備考
2013/02	1.0	初版
2013/06	1.1	「−−no_all」オプションの追加 <cond>ノードの比較演算子の変更
2014/03	1.2	template ファイル記述方式変更 対応ソルバの追加
2014/10	1.3	ワークフローの Lua 化に対応
2014/10	1.3.1	スナップショットファイル名を.pdi_params から snap_params.pdi に変更
2015/08	1.4	JAXA の多目的進化計算 (MOEA) に対応
2015/09	1.4.1	JAXA の多目的進化計算 (MOEA) を Windows 環境に対応 Windows 環境用に py2exe の setup.py を提供
2015/09	1.4.2	プリファレンスファイルが存在しない場合にデフォルト設定を使用
2015/10	1.4.3	Xjob(hpcpfGUI) の改版に対応
2015/10	1.4.4	生成する Lua スクリプトの構成を変更

目 次

はじめに

本書は、HPC/PF システムの PDI サブシステムについての操作説明書です。

HPC/PF システムは、数値解析シミュレータとその周辺処理ツール群であり、HPC/PF システムを構成するサブシステムの１つである PDI は、ユーザのパラメータ空間設計およびパラメータ入力を支援します。

本書では、PDI のインストール方法および操作方法について説明します。

1　インストール

PDI のインストール方法について説明します。

1.1　動作環境

想定される動作環境を以下に示します。

表 2-1 PDI 動作環境

アーキテクチャ	AMD64, IA32
OS	Linux 2.6/3.x MacOS 10.10/10.11 Windows 7/8.1
ソフトウエア	Python 2.x wxPython 2.7 以降/3.0

1.2　必須ソフトウエアのインストール

1.2.1　Python のインストール

(1) Linux

多くの Linux ディストリビューションでは、Python の実行環境は OS に含まれています。ログイン後にターミナルで「`which python`」と入力し、Python の実行パスが表示されれば、そのシステムで Python は使用可能です。

Python がインストールされていない場合、各 Linux のディストリビューション管理システムを使用して Python をインストールすることが可能です。以下に例を示します。

RedHat/CentOS 系

```
sudo yum install python
```

Debian/Ubuntu 系

```
sudo apt-get install python
```

(2) MacOS

MacOS 環境向けには、Python のインストーラパッケージが用意されています。

以下に示す URL より、PDI を動作させる環境にあったインストーラをダウンロードし、実行することでインストールをおこなってください。

```
http://www.python.org/download/
```

尚、同 URL には Python 3.x のインストーラパッケージも登録されていますが、PDI は Python 3.x での動作は保証していません。Python 2.x のインストーラパッケージを使用してください。

(3) Windows

Windows 環境向けには、Python 2.7.x のインストーラパッケージが用意されています。

以下に示す URL より、PDI を動作させる環境にあったインストーラをダウンロードし、実行することでインストールをおこなってください。

```
http://www.python.org/download/
```

尚、同 URL には Python 3.x のインストーラパッケージも登録されていますが、PDI は Python 3.x での動作は保証していません。Python 2.x のインストーラパッケージを使用してください。

インストーラ実行時に、「Customize Python 2.7.x」という画面が表示され、ここでインストールする Python の構成を選択する事が出来ます。

図 1 Python Windows インストーラ画面

ここで、デフォルトでは「Add python.exe to Path」という項目が「×」(インストールしない) になっているので、これを「Will be installed on local hard drive」に変更した上でインストールを進めてください。

1.2.2 wxPython のインストール

(1) Linux

多くの Linux ディストリビューションでは、ディストリビューション管理システムを使用して wxPython をインストールすることが可能です。以下に例を示します。

RedHat/CentOS 系

```
sudo yum install wxPython
```

Debian/Ubuntu 系

```
sudo apt-get install python-wxgtk2.8
```

(2) MacOS

MacOS 環境向けには、wxPython のインストーラパッケージが用意されています。以下に示す URL より、PDI を動作させる環境にあったインストーラをダウンロードし、実行することでインストールを行ってください。

```
http://www.wxpython.org/download.php#stable
```

(3) Windows

Windows 環境向けには、wxPython のインストーラパッケージが用意されています。以下に示す URL より、PDI を動作させる環境にあったインストーラをダウンロードし、実行することでインストールをおこなってください。

```
http://www.wxpython.org/download.php#stable
```

1.3 PDI のインストール

PDI のパッケージは、システムの任意の場所に置くことができます。

PDI のディストリビューションに含まれる pdi_*version*.tar.gz または pdi_*version*.zip ファイル (*version* は実際の文字列に置き換えてください) を任意のディレクトリに展開します。

tar.gz の場合は、tar コマンドで以下のように実行します。

```
tar xvfz pdi_version.tar.gz
```

展開を実行すると、以下に示すようなディレクトリ階層が作成されます。

```
pdi
├──── bin
│     ├──── pdi
│     └──── pdi.bat
```

```
├──── doc
│     ├──── pdi_ug.pdf
│     └──── pdi.txt
├──── lib
│     ├──── cheetah
│     └──── python
└──── conf
      ├──── PDI.conf
      └──── PDI_log.conf
```

PDI の実行コマンドは、`pdi/bin/pdi`(Linux, MacOS) または `pdi/bin/pdi.bat`(Windows) です。

2 操作方法

PDI の操作方法について説明します。

2.1 プログラムの起動

PDI の実行コマンドファイルは、以下のパスに存在します。

(PDI インストールディレクトリ)/pdi/bin/pdi(.bat)

PDI を起動する際の、コマンドライン指定の形式を以下に示します。。

GUI モード

```
pdi  [-x case_directory] [-d param_desc] [-o output_pattern]
     [-t template_file -t template_file ...] [--no_all]
     [-p param_name:param_value  -p param_name:param_value ...]
```

バッチモード

```
pdi {-b | -B} [-x case_directory] [-d param_desc] [-o output_pattern]
    [-t template_file -t template_file ...]
    [-p param_name:param_value -p  param_name:param_value ...]
```

引数説明

-b
: バッチモード実行。指定された場合、GUI ウインドウを表示せず、バッチモードで実行します。

-B
: バッチモード実行。-b 指定の場合と同じですが、ワークフロー実行用 Lua スクリプトファイル (cwf.lua および pdi_generated.lua) の生成は行いません。

-x case_directory
: ケースディレクトリ指定。指定されると、PDI は起動時に case_directory にカレントワーキングディレクトリを移動します。指定が省略された場合は、起動ディレクトリとなります。

-d param_desc
: パラメータ記述 XML ファイルの指定。パラメータ記述 XML ファイルのパスを指定します。絶対パスか、カレントワーキングディレクトリからの相対パスで指定します。

--no_all
: GUI モードにおいて「_All_」ページを作成しません。ただし、パラメータ記述ファイル中に＜ group ＞タグが一つも存在しない場合は作成します。

-t template_file
: ソルバ入力パラメータテンプレートファイルの指定。入力パラメータテンプレートファイルのパスを指定します。絶対パスか、カレントワーキングディレクトリからの相対パスで指定します。複数指定が可能です。

`-o output_pattern`

出力先ディレクトリ・ファイル名のパターン指定。サブケース毎に生成する作業ディレクトリ及び作業ディレクトリ配下の入力パラメータファイルのパターンを指定します。パターンは、以下に示す形式で指定します。

```
directoryname/filename
```

作業ディレクトリ名と配下のファイル名は「/」で区切られます。パターン中に「/」が現れない場合、すべて filename と解釈され、カレントワーキングディレクトリ直下に filename のパターンで入力パラメータファイルが作成されます。

directoryname が指定された場合、カレントワーキングディレクトリ配下に作業ディレクトリとして directoryname が作成され、その配下に filename が作成されます。

directoryname 中に「%P」が含まれている場合、この部分はスイープされるパラメータ値の組み合わせ文字列に置き換えられます。また、「%Q」がが含まれている場合、この部分はスイープされるパラメータ名+パラメータ値の組み合わせ文字列に置き換えられます。

```
(例)
ptest%P/PARAMS   ⇒   ptest_100_10/PARAMS
ptest%Q/PARAMS   ⇒   ptest_Re100_CX10/PARAMS
```

パラメータスイープが行われない場合は、「%P」「%Q」は空文字列に置き換えられます。

```
(例) ptest%P/PARAMS   ⇒   ptest/PARAMS
```

filename 中に「%T」が含まれている場合、この部分は template_file のベース名に置き換えられます。また、filename 中に「#T」が含まれている場合、この部分は template_file の番号に置き換えられます (template_file が 1 個だけ指定されている場合は「_0」)。

filename 中に「#S」が含まれている場合、この部分は単一ディレクトリ内のサブケース番号 (1 ディレクトリにつき 1 サブケースの場合は、常に 0) に置き換えられます。

directoryname および filename 中に「#D」、「#J」が含まれている場合、この部分はそれぞれディレクトリ通番、サブケース通番に置き換えられます。

```
(例) ptest#D/PARAMS#J   ⇒   ptest_0/PARAMS_0
```

`-p param_name:param_value`

パラメータ値の直接指定。パラメータ名とパラメータ値のペアを、コマンドラインから直接指定します。複数指定が可能です。パラメータ名とパラメータ値は「:」で区切って記述します。

パラメータ値は、以下のいずれかの形式で指定を行います。

- 直接指定
- 範囲、刻み幅指定： 最小値/最大値/刻み幅
  最小値、最大値、刻み幅を「/」で区切って記述。空白を入れてはならない。
- 列挙： 値 1, 値 2,...
  列挙する値を「,」で区切って記述。空白を入れてはならない。

尚、直接指定または列挙で文字列値を指定する場合、空白または区切り文字が含まれる場合は「"」で囲って指定します。

PDI を GUI モードで実行すると、PDI のメインウインドウが表示されます。

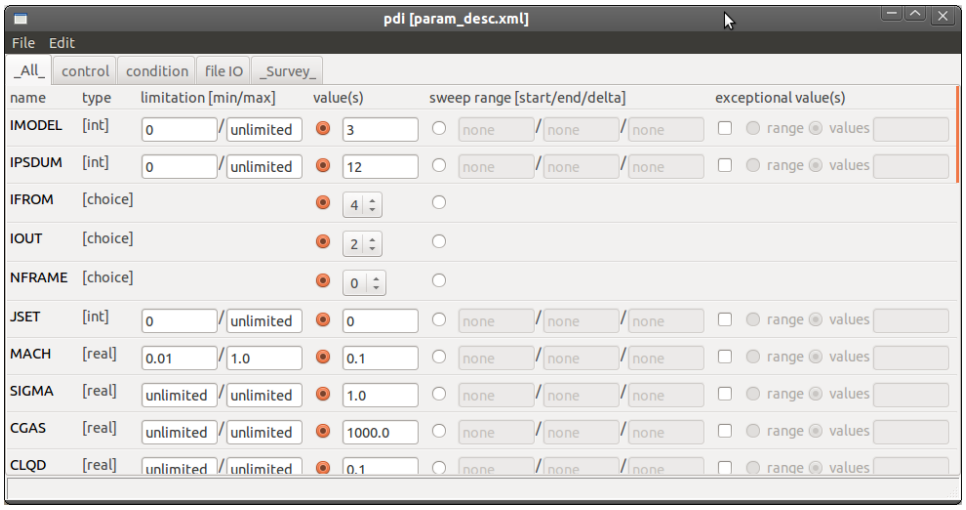

図 2 PDI メインウインドウ

尚、PDI の終了は、メインウインドウの File メニューから Quit を選択するか、メインウインドウを閉じることで行えます (MacOS の場合、Quit はメニューバーの「Python」メニューの「`Quit Python`」です)。

2.2　パラメータ空間設計支援機能

PDI のパラメータ空間設計支援は、パラメータ定義ファイルの記述に基づいて行われます。

パラメータ定義ファイルは、PDI で設定可能なソルバのパラメータの名前、型、値域、デフォルト値、所属グループなどを記述した XML 形式のファイルで、PDI の起動時に「-d」オプションで指定する他、GUI モードの場合は「File」メニューの「`load parameter description file`」からファイルを指定し、ロードさせることができます。

パラメータ定義ファイルがロードされると、GUI モードの場合パラメータ定義ファイル中に記述された各パラメータ項目は、所属するグループごとにタブページにまとめられ、リスト形式で表示されます。所属グループ属性のないパラメータ項目を含めすべてのパラメータ項目は、「ALL」のタブページにまとめられます。

各パラメータ項目は、パラメータの型に応じてパラメータ値および範囲と刻み幅の入力欄の GUI 部品が配置されます。

name パラメータ名
: パラメータ定義ファイル中にパラメータの説明が記述されている場合、マウスカーソルをパラメータ名の上に移動するとパラメータの説明がツールチップとして表示されます。また、パラメータ名をマウス左ボタンでダブルクリックすると、そのパラメータ設定の有効／無効が切り替わります。無効の場合はソルバ入力ファイル生成時にそのパラメータの設定は無視され、テンプレートファイルに設定されたデフォルト値が使用されます。パラメータ設定が有効で、かつそのパラメータ空間が縮退している (有効なパラメータ設定値が無い) 場合は、パラメータ名は赤色で表示されます。

type パラメータ型
: パラメータ定義ファイル中に記述されたパラメータの型が表示されます。

- int 整数
- real 実数
- choice 候補選択
- string 文字列
- bool 真偽値

limitation 値域
パラメータ値が取りうる値の最小／最大値を設定します。制限なしにする場合には、none または unlimited と入力します。type が int または real の場合のみ設定可能です。

value(s) パラメータ値 (単一または列挙)
このラジオボタンが ON の場合には、パラメータ値は直接指定します。type が int または real の場合は、空白で区切って複数の値を列挙指定することができます。

sweep range パラメータスイープ範囲
このラジオボタンが ON の場合には、パラメータ値は範囲と刻み幅で指定します。type が bool の場合は、パラメータ値は True と False の両方でスイープします。type が choice の場合は、全選択肢がパラメータスイープの対象になります。type が string のパラメータは、パラメータスイープの対象にはなりません。

exceptional value(s) 除外値 (範囲または列挙値)
このチェックボックスが ON の場合、パラメータスイープの除外値を設定することができます。range ラジオボタンが ON の場合は、テキストボックスに除外する範囲の最小値と最大値を「/」(スラッシュ) で区切って指定します。values ラジオボタンが ON の場合は、テキストボックスに除外する値を直接指定します。複数の値を除外する場合は、空白で区切って列挙指定することもできます。

尚、除外値の設定を行うことができるのは、type が int または real のパラメータ項目のみです。

設定の例

`limitation -0.3 / 0.7` (`-0.3` から `0.7` までの値が有効)

`sweep range -0.5 / 0.5 / 0.1` (`-0.5` から `0.5` まで `0.1` 刻みでパラメータスイープ)

`exceptional values 0.0 0.1` (`0.0` と `0.1` はパラメータスイープから除外)

パラメータスイープされる値： `-0.3, -0.2, -0.1, 0.2, 0.3, 0.4, 0.5`
(`-0.4, -0.5` は `limitation` から外れるので無効、`0.0` と `0.1` は除外)

パラメータ間の依存関係

パラメータ定義ファイル中に、パラメータ間の依存関係 (<depend>タグ) の記述がある場合は、依存先パラメータの値に応じてパラメータ項目欄の状態の変更（有効/無効、値設定、値域設定）が行われます。

ただし、この状態変更は依存先のパラメータ値が変更された際に行われるので、その後パラメータ欄の操作を行うことでパラメータの状態を再変更することが可能です。

上記の設定において、複数のパラメータ値をとるように設定されたパラメータ項目が 1 個でも存在する場合はパラメータスイープの対象となります。パラメータスイープの総件数は、すべてのパラメータ項目のパラメータ値の件数を掛け合わせた数になります。

2.3 パラメータサーベイ方式設定機能

GUI モードで起動された PDI の「Survey」タブを選択すると、下図に示すパネルが表示され、ここでパラメータサーベイの方式を設定することができます。

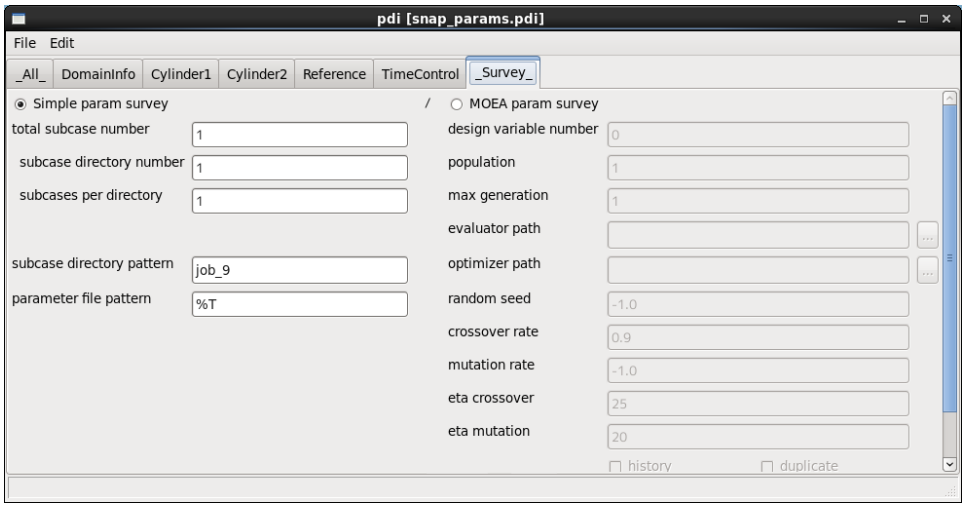

図 3 PDI の Survey パネル

「Survey」タブ最上段のラジオボタンで「Simple param survey」が選択されている場合、「Survey」タブの左側が有効化されており、以下に示す項目を設定する事ができます。

全てのパラメータケース (サブケース) を組み合わせた数 (全探査数) が「`Total subcase number`」に表示されます。この数が 0 の場合は、パラメータ空間が縮退していることを意味し、ソルバ入力用のパラメータファイル生成を行うことはできません。

全探査数に対して、幾つの作業ディレクトリを作成するかを「`subcase directory number`」に、または 1 つの作業ディレクトリにつき幾つのパラメータケースを割り当てるかを「`subcases per directory`」に入力します。どちらか一方が入力されると、もう一方は自動的に更新されます。初期値は、1 つの作業ディレクトリにつき 1 つのパラメータケース (作業ディレクトリ数=全探査数) です。(作業ディレクトリ数) × (作業ディレクトリ当りのパラメータケース数) ≠ (全探査数) の場合は、最後の作業ディレクトリに割り当てられるパラメータケース数で調整されます。

作業ディレクトリ名 (`subcase directory pattern`)、ソルバ入力用パラメータファイル名 (`parameter file pattern`) については、以下に示すキーワードを使用してパターンを設定します。

%P パラメータ組み合わせ

%Q パラメータ組み合わせ (パラメータ名付き)

%T テンプレートファイルのベース名
(テンプレートファイル名から末尾の".template", ".tmpl", ".tmp", ".tpl"を 除いたもの)

#D 作業ディレクトリ通番

#J サブケース通番

#T ソルバ入力パラメータファイルが複数ある場合のファイル番号

尚、「Survey」タブ最上段のラジオボタンで「MOEA param survey」を選択すると、「Survey」タブの右側部分が有効になり、MOEA(多目的進化計算) による最適化の設定を行う事ができます。

PDI による MOEA 最適化については「3 MOEA 最適化の設定」を参照して下さい。

2.4　ソルバ入力用パラメータファイルの生成機能

PDI によるソルバ入力用パラメータファイルの生成を行うには、パラメータ空間の設定とパラメータサーベイ方式の設定に加え、ソルバ入力パラメータテンプレートファイルの指定と、ソルバ種別の指定が必要です。

2.4.1　ソルバ入力パラメータテンプレートファイルの指定

ソルバ入力パラメータテンプレートファイルは、PDI の起動時に「`-t`」オプションで指定することができます。「`-t template_file`」という形式で、複数指定することが可能です。

PDI を GUI モードで起動した場合、「Edit」メニューの「`set parameter template file(s)`」から、ソルバ入力用パラメータテンプレートファイルの追加／削除を行うことができます。

このメニューを選択すると、下図に示すダイアログウインドウが表示されます。

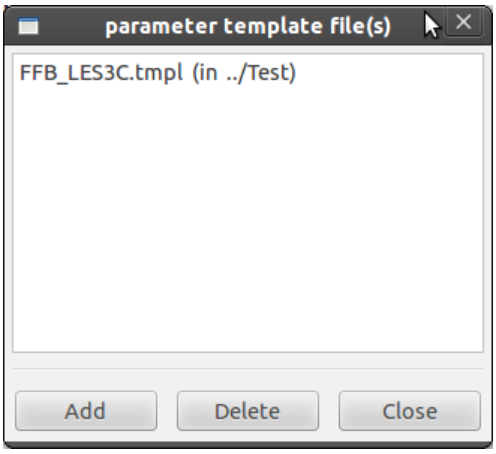

図 4 PDI のテンプレートファイル設定ダイアログ

このダイアログウインドウには、既に登録済みのソルバ入力用パラメータテンプレートファイルがリスト表示されています。このダイアログは「Close」ボタンをクリックすると閉じられます。

「Add」ボタンをクリックすると、ファイル選択ダイアログが表示され、ここで選択したファイルがソルバ入力用パラメータテンプレートファイルととして追加されます。

リスト上で登録済みのソルバ入力用パラメータテンプレートファイルを選択状態にし、「Delete」ボタンをクリックすると、そのファイルの登録は解除されます。

2.4.2　ソルバ種別の設定

PDI では、ソルバ入力用パラメータファイルの生成時に、パラメータスイープ実行のための設定ファイルを生成します。この設定ファイル中には、ソルバ実行の方法の記述も含まれるため、PDI

上でソルバ種別の設定を行なっておく必要があります。

GUI モードで起動した PDI の「Edit」メニューから「select solver type」を選択すると、下図に示すダイアログウインドウが表示されます。

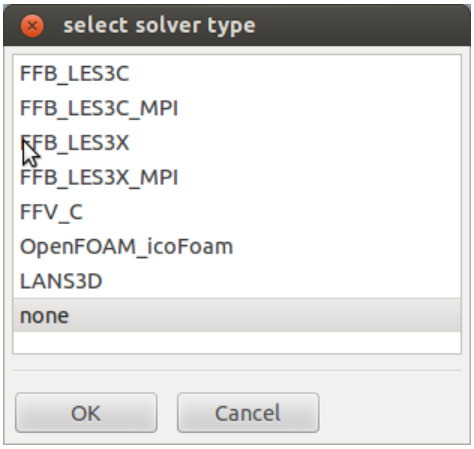

図 5 PDI のソルバ種別選択ダイアログ

ここには、PDI に予め登録されているソルバ種別の一覧がリスト表示され、リスト中の項目を選択状態にして「OK」ボタンをクリックすると、ソルバ種別の設定がおこなわれます。

現在、PDI に登録されているソルバ種別の設定を以下に示します。

ソルバ種別	ソルバ実行コマンド	ソルバ入力パラメータファイル名パターン
FFB_LES3C	run_les3c.sh	PARMLES3C
FFB_LES3C_MPI	run_les3c_mpi.sh	PARMLES3C
FFB_LES3X	run_les3x.sh	PARMLES3X
FFB_LES3X_MPI	run_les3x_mpi.sh	PARMLES3X
FFV	run_ffv.sh %T	%T
OpenFoam_icoFoam	run_openfoam.sh	
その他 (none)	run.sh	変更せず

上記の表中の「%T」は テンプレートファイルのベース名を表します。

また、パラメータスイープ実行のための設定ファイルにおいては、上記の表の「ソルバ実行コマンド」に示された実行ファイル (シェルスクリプト) がケースディレクトリ配下に存在することを前提としています。

2.4.3　サブケースディレクトリとソルバ入力パラメータファイルの生成

GUI モードで起動された PDI では、「File」メニューの「generate solver parameter file(s)」を選択すると、パラメータスイープ用の作業ディレクトリ (サブケースディレクトリ) と、ソルバ入力用パラメータファイルの生成が行われます。

具体的には、以下に示す処理が行われます。

(1) サブケースディレクトリの作成

ケースディレクトリ配下に、作業ディレクトリ名パターンを展開したディレクトリが「subcase directory number」個作成されます。作成しようとしているディレクトリが既に存在する場合は、そのディレクトリはそのまま流用されます。

(2) ソルバ入力用パラメータファイルの作成

作成したサブケースディレクトリの配下に、ソルバ入力用パラメータファイル名パターンを展開したファイル名で、ソルバ入力用パラメータファイルが作成されます。PDI は、指定されているソルバ入力パラメータテンプレートファイル中のマクロ記述をパラメータ値に置換することで、ソルバ入力用パラメータファイルを生成します。複数のテンプレートファイルが指定されている場合、それらすべてに対応するソルバ入力用パラメータファイルが作成されます。

作成しようとしているソルバ入力用パラメータファイルが既に存在する場合は、そのファイルは上書きされます。

(3) ワークフロー実行用パラメータ設定ファイルの作成

実際のソルバ実行を含むパラメータスイープ／パラメータサーベイの実行は、ケースワークフローから Lua スクリプトとして実行されます。

PDI は、ソルバ入力用パラメータファイルの生成時に、Lua スクリプトファイルとして、ケースディレクトリ配下に pdi_generated.lua というファイルを作成します。同名のファイルが既に存在する場合、そのファイルは上書きされます。

pdi_generated.lua は、ケースワークフローファイル (cwf.lua) から require され、関数 GET_JOBS(...) を呼び出し、戻り値としてサブケースジョブ群が格納されたテーブルを受け取ることを前提として記述されています。

尚、PDI はケースディレクトリ配下に cwf.lua が存在しない場合は、以下の内容の cwf.lua ファイルを作成します。

```
require('hpcpf')
local ex = ...
local caseDir = ex.caseDir
print('Case dir = ' .. caseDir)
local dxjob = require('dxjob')
local dj = dxjob.new(ex)
require('pdi_generated')
jobs = GET_JOBS(ex)
for k, j in pairs(jobs) do
  dj:AddJob(j)
end
dj:GenerateBootSh()
dj:SendCaseDir()
dj:SubmitAndWait()
dj:GetCaseDir()
return ex.outputFiles
```

cwf.lua が存在している場合は、cwf.lua を上書きするかどうかを尋ねるダイアログが表示されます。ただし、「-B」オプションでバッチモードで PDI を実行した場合は、cwf.lua も pdi_generated.lua も作成されません。

(4) パラメータリストファイルの作成

パラメータリストファイルは、現在のパラメータ空間設定の下で実行されるパラメータスイープのパラメータ値を記述した CSV 形式のテキストファイルです。

PDI は、ソルバ入力用パラメータファイルの生成時に、パラメータリストファイルをケースディレクトリ配下に param_list.csv というファイル名で作成します。同名のファイルが既に存在する場合、そのファイルは上書きされます。

2.5 パラメータ空間設定情報の保存

PDI は終了時に、その時点でのパラメータ空間設定情報とパラメータサーベイ方式設定情報を、ケースディレクトリ配下に snap_params.pdi というファイル名で保存します。既にこのファイルが存在する場合、ファイルは上書きされます。

snap_params.pdi ファイルが存在するディレクトリがケースディレクトリとして指定 (-x) された場合、パラメータ記述ファイルの指定 (-d) がなくても PDI はパラメータ空間設計を行うことが可能です (この場合、ログにはワーニングが出力されます)。また、パラメータ記述ファイルの指定が行われた場合は、パラメータ記述ファイルのロード後に snap_params.pdi ファイルに保存された情報の反映を行います。

ただし、指定されたパラメータ記述ファイルと、snap_params.pdi ファイルに保存されているパラメータ記述ファイル名が異なる場合は、snap_params.pdi ファイルの内容は無視されます。

2.6 バッチモードでの PDI 実行

PDI 起動時に「-b」または「-B」オプションを指定すると、PDI は GUI ウインドウを表示せず、バッチモードで実行されます。

バッチモードでの PDI 実行は主に、以前に設定されたパラメータスイープにおけるパラメータ空間の変更を行うために行います。PDI 起動時に「-p」オプションで 1 個ないし複数個のパラメータ値を変更することにより、ケースディレクトリ/snap_params.pdi ファイルに保存された前回設定のパラメータ空間が更新されます。ただしこの場合は、前回パラメータ空間を設定した際と同じケースディレクトリが指定されていることが必要です。

実行の例

```
pdi -x case01 -p Re:100/200/10
```

(ケースディレクトリ「case01」において、「Re」という名前のパラメータを、100 から 200 まで 10 刻みでスイープさせる)

PDI の「-p」オプション指定方法の詳細は、2.1 章を参照してください。

3 MOEA 最適化の設定

3.1 概要

工学的な最適化問題は、最適化の対象である目的関数が複数 (一般的には 2 個以上) 存在する多目的最適化問題である場合が多く、しかもそれらは相反する性質を持つ場合も珍しくありません。

JAXA(宇宙航空研究開発機構 [1]) が開発した Cheetah は、生物の進化のメカニズムを模倣した進化的アルゴリズム (EA) を多目的最適化問題に適用する MOEA(多目的進化計算) のソフトウエアフレームワークです。

PDI では、設定したパラメータ空間内での最適化計算を Cheetah を使用して実行するためのスクリプトの生成と、実行環境を設定する機能を含んでいます。

本章では、PDI による MOEA のパラメータ設定と実行方法について説明します。

尚、JAXA による多目的設計探査に関する詳細については、以下の URL を参照して下さい。

```
http://flab.eng.isas.jaxa.jp/monozukuri/mode/index.html
```

3.2 MOEA の設定方法

MOEA の設定は「Survey」タブを選択し、最上段の「MOEA param survey」ラジオボタンを選択する事で行えるようになります。選択すると「Survey」タブ上で有効化されている部分が変更され、MOEA の設定が可能になります。

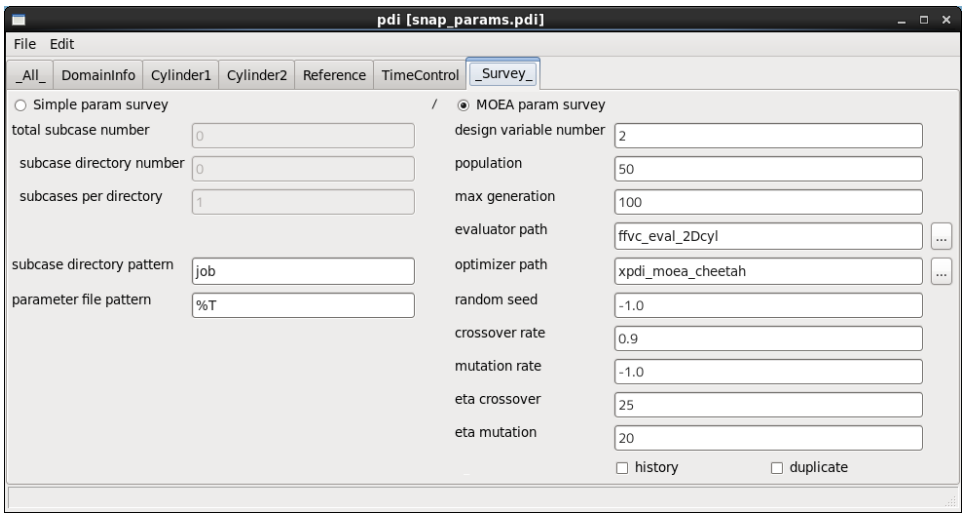

図 6 PDI の Survey パネル (「MOEA param survey」選択時)

ここでは、以下に示す項目を設定します。

subcase directory pattern

作業ディレクトリ名のパターン (ベース部分) を指定します。「Simple param survey」選択時と指定方法は同じですが、パターン表現用のキーワード (%P, %Q, #D 等) は全て空文字

[1] http://www.jaxa.jp

列と置き換えられ (取り除かれ)、残った文字列を作業ディレクトリ名のベース部分として使用します。

実際の作業ディレクトリは、ケースディレクトリ配下に「ベース部分_整数」という形式で、population(後述) 個作成されます。

parameter file pattern

各作業ディレクトリ配下に作成される、ソルバ入力用パラメータファイルのファイル名パターンを指定します。「Simple param survey」選択時と指定方法は同じです。

design variable number

MOEA における設計変数となるパラメータ項目の数が表示されます。この値は変更できません。

PDI における MOEA param sweep では、「レンジ指定」を選択し、範囲 (最小/最大値) を設定した int 型または real 型のパラメータを設計変数として扱います。「design variable number」欄には、この条件を満たすパラメータ項目の数が表示されます。

この欄の値が 0 の場合は、探索すべき設計変数が存在しない事を意味し、MOEA 計算は行えません。

尚、「Survey」タブで「MOEA param sweep」を選択した場合、各パラメータ項目設定タブ上では「レンジ指定」を表す「sweep range」の表記が「search range」に変わり、パラメータの刻み幅入力欄 (delta) は有効精度の入力欄 (precision) に変わります (下図参照)。

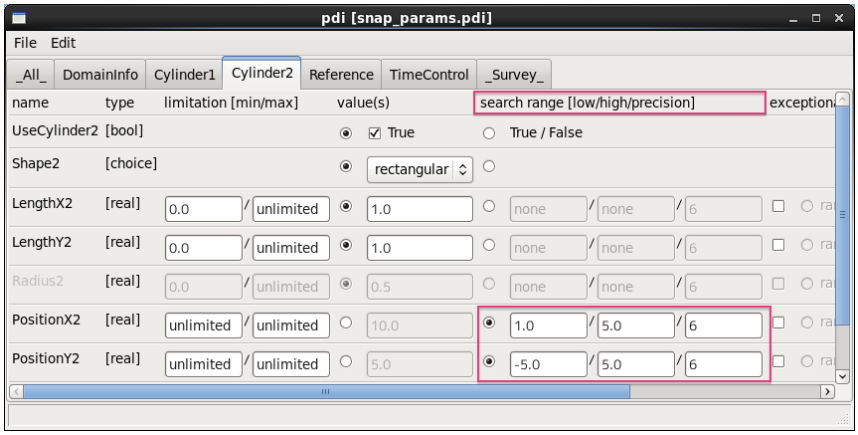

図 7 PDI のパラメータ設定パネル (「MOEA param survey」選択時)

population

MOEA 計算における個体数を設定します。ここで設定した個数だけ作業ディレクトリが作成されます。4 の倍数である必要があります。また、0 以下の値は設定できません。

max generation

MOEA 計算における世代数を設定します。ここで設定した回数だけ

{ ソルバー実行 ⇒ パラメータ値 (設計変数) の進化計算 }

がループ実行されます。0 以下の値は設定できません。

evaluator path
MOEA 計算における評価モジュール (evaluator) のパスを設定します。入力欄の右の「...」ボタンをクリックするとファイル選択ダイアログが表示され、ここでファイルを選択すると入力欄に反映されます。

評価モジュールは、ソルバーの実行結果から目的関数の値を計算する外部プログラムであり、最適化問題毎に使用者が用意する必要があります。評価モジュールの仕様 (インターフェースの詳細) については、「5 外部プログラムの形式」を参照して下さい。

optimizer path
MOEA 計算における進化計算モジュール (optimizer) のパスを設定します。入力欄の右の「...」ボタンをクリックするとファイル選択ダイアログが表示され、ここでファイルを選択すると入力欄に反映されます。

進化計算モジュールは、目的関数の値から、次の世代の設計変数を生成する外部プログラムであり、PDI では Cheetah を使用した進化計算モジュール (`xpdi_moea_cheetah`) を提供しています。進化計算モジュールの仕様 (インターフェースの詳細) については、「5 外部プログラムの形式」を参照して下さい。

random seed
MOEA 計算において使用する乱数の種を設定します。有効な値は 0.0〜1.0 ですが、負値を設定した場合は乱数を用いて設定されます。

crossover rate
MOEA 計算における交叉率を設定します。有効な値は 0.0〜1.0 です。

mutation rate
MOEA 計算における突然変異率を設定します。有効な値は 0.0〜1.0 ですが、負値を設定した場合は自動的に (1.0/設計変数個数) に設定されます。

eta crossover
MOEA 計算における交叉の distribution index を設定します。有効な値は 5〜40 です。

eta mutation
MOEA 計算における突然変異の distribution index を設定します。有効な値は 5〜50 です。

history
MOEA 計算において、全評価個体の履歴ファイル (interface/history.txt) を作成するかどうかを設定します。

duplicate
MOEA 計算において、重複する設計点を作成するかどうかを設定します。

3.3 MOEA 実行環境の生成

前節に示した MOEA の設定を行い、「File」メニューの「`generate solver parameter file(s)`」を選択すると、MOEA 計算実行用の環境が生成されます。

ケースディレクトリ配下に以下のようなディレクトリ・ファイルが生成されます。

```
[ケースディレクトリ]
  ├──── (cwf.lua)           (ケースワークフローファイル)
  ├──── (input_data)
  │       └──── (run.sh)    (ソルバー実行スクリプト)
  ├──── cfg
  │       ├──── moea.cfg    (Cheetah 用コンフィグレーションファイル)
  │       └──── mop.cfg     (目的関数定義ファイル)
  ├──── job_{0 〜 個体数-1} (サブケースディレクトリ)
  │       └──── run.sh      (ソルバー実行スクリプト)
  ├──── interface           (Cheetah 作業用ディレクトリ)
  ├──── moea_out            (Cheetah 出力用ディレクトリ)
  └──── pdi_generated.lua   (ワークフロー記述 Lua ファイル)
```

ソルバー実行スクリプトは、input_data ディレクトリ配下から、全てのサブケースディレクトリ配下にコピーされます。ファイル名はソルバー種別によって異なります。

ワークフロー記述 Lua ファイル (pdi_generated.lua) は、同時に生成されるケースワークフローファイル (cwf.lua) から require され、関数 GET_JOBS(...) を呼び出し、戻り値としてサブケースジョブ群が格納されたテーブルを受け取ることを前提として記述されています。

尚、input_data ディレクトリは、ケースディレクトリに事前に用意されている事が前提です。

3.4 MOEA の実行

PDI で生成した MOEA 実行環境は、ケースディレクトリでケースワークフローを実行する事で実行されます。この実行には、Xjob(hpcpfGUI に付属) のライブラリパスが環境変数 LUA_PATH に設定されている事が必要です [2]。

```
$ cd "ケースディレクトリ"
$ export LUA_PATH="hpcpfGUI インストールディレクトリ"/lib
$ lua cwf.lua
```

以下に、ケースワークフローが実行された際に行われる MOEA 計算処理の概要を示します。

1. optimizer(xpdi_moea_cheetah) を世代番号 0 で実行します。これにより、探査空間内のランダムな設計点が個体数個生成され、interface/pop_vars_eval.txt ファイルに出力されます。
2. 内部コマンド xpdi_genparam を実行します。これにより、内部的に PDI がバッチモードで実行され、各サブケースディレクトリ配下にそるバー入力パラメータファイルが生成されます。
3. 個体数個分のソルバー実行ジョブが発行されます。
4. 全てのジョブの完了後、evaluator が実行されます。これにより、各ジョブ毎の目的関数 (および制約条件) が計算され、interface/pop_objs_eval.txt ファイル (および pop_cons_eval.txt) に出力されます。

[2] hpcpfGUI から実行される場合は、ディレクトリ移動と環境変数設定は自動で行われています。

5. 世代番号を +1 し、最大世代数回、処理 1 から繰り返します。

計算結果は、moea_out/allpop.txt に出力されます。

4 ファイルフォーマット

4.1 パラメータ定義ファイル

パラメータ定義ファイルは、XML 形式のテキストファイルとして記述されます。

以下に、構成要素となる XML ノードの記述方式を示します。

(1) トップレベルノード

パラメータ定義ファイルのトップレベルノードは<hpcpf_paramdesc>です。

このノード配下に記述された内容だけが PDI の処理対象となります。

[記述形式]

```
<hpcpf_paramdesc [classification="snapshot"]>
...
</hpcpf_paramdesc>
```

<hpcpf_paramdesc>ノードには、classification 属性としてキーワード"snapshot"を記述することができます。このキーワードが記述されたパラメータ定義ファイルは、パラメータ空間設定情報を保存したスナップショットデータファイルと認識されます。スナップショットデータファイルは通常、PDI の終了時にケースディレクトリ配下に snap_params.pdi というファイル名で作成されます。

(2) パラメータ記述ノード

パラメータ項目の記述には、<param>ノードを使用します。

[記述形式]

```
<param name="パラメータ名" type="パラメータタイプ" [disable="True|False"]>
  { パラメータ説明文 }
  { サブノード群 }
</param>
```

<param>ノードには、name 属性および type 属性を記述する必要があります。

name 属性には、パラメータ名を記述します。

type 属性には、以下のいずれかのパラメータタイプを記述します。

int 整数

real 実数

bool 真偽値 (True | False)

string 文字列

choice 文字列候補

<param>ノードには、disable 属性を記述することができます。disable 属性には"True"または"False"のキーワードを記述し、True の場合はそのパラメータは無効化されます。記述を省略した場合、disable 属性は"False"とみなされます。

<param>ノード配下には、パラメータ説明のテキストと、複数のサブノードを記述することができます。

<param>ノード配下の最初のテキストは、パラメータ説明の記述として扱われます。

<param>ノード配下に記述できるサブノードについては、以下の説明を参照して下さい。

(3) パラメータ記述サブノード

以下は、<param>ノード配下に記述できるサブノードの説明です。

(3-a) <item>ノード

<item>ノードは、<param>ノードの type 属性が"choice"の場合の、文字列候補の 1 つを記述するノードで、<param>ノード配下に複数 (選択候補数) 個記述できます。

[記述形式]

```
<item>文字列候補 1</item>
<item>文字列候補 2</item>
```

<param>ノードの type 属性が"choice"意外の場合は、<item>ノードの記述は無視されます。

(3-b) <range>ノード・<minmax>ノード

<range>ノードおよび<minmax>ノードは、<param>ノードの type 属性が"int"または"real"の場合の、値域を記述するノードです。

[記述形式]

```
<range min="最小値" max="最大値" />
<minmax min="最小値" max="最大値" />
```

min 属性に値域の最小値を、max 属性に最大値を記述します。min 属性または max 属性の記述を省略した場合、省略された方向の値制限は無しになります。また、<range>ノードおよび<minmax>ノードの記述自体を省略すると、値域制限は無しになります。

(3-c) <value>ノード

<value>ノードは、パラメータ値の初期値を記述するノードです。

[記述形式]

```
<value>パラメータ値</value>
```

<value>ノードの記述を省略した場合、パラメータ値の初期値は<param>ノードの type 属性に応

じ、以下に示すようになります。

int "0"

real "0.0"

bool "False"

string "" (空文字列)

choice "0"(先頭の候補)

<param>ノードの type 属性が"int"または"real"の場合は、パラメータ値は空白で区切って複数の値を列挙することができます。

(3-d) <useRange>ノード

<useRange>ノードは、<param>ノードのパラメータ値を値域・刻み幅指定で定義するかどうかを記述するノードです。

[記述形式]

```
<useRange> True | False </useRange>
```

"False"の場合、<value>ノード記述に従ってパラメータ値を設定します。

"True"の場合、<value>ノードの記述は参照せず、次項で説明する<sweepRange>ノードの記述に従ってパラメータ値を設定します。ただし、<param>ノードの type 属性が"bool"の場合はパラメータ値が"True"と"False"の 2 サブケースとして、<param>ノードの type 属性が"choice"の場合は<item>ノードで記述した全ての候補をパラメータ値とするサブケースが設定されます。<param>ノードの type 属性が"string"の場合、<useRange>ノードの値は無視され、常に<value>ノードの記述に従ってパラメータ値が設定されます。

<useRange>ノードの記述を省略した場合のデフォルト値は"False"です。

(3-e) <sweepRange>ノード

<sweepRange>ノードは、<useRange>ノードの値が"True"の場合の、値域・刻み幅指定によるパラメータ値の設定を記述するノードです。

[記述形式]

```
<sweepRange min="最小値" max="最大値" delta="刻み幅" />
```

min 属性に値域の最小値を、max 属性に最大値を、delta 属性に刻み幅を記述します。これらの属性の記述は省略できません。

尚、<sweepRange>ノードの記述は<param>ノードの type 属性が"int"または"real"の場合以外は無視されます。

(3-f) <useExcept>ノード

<useExcept>ノードは、<param>ノードのパラメータ値に対する除外値指定を参照するかどうかを記述するノードです。

[記述形式]

```
<useExcept> True | False </useExcept>
```

"False"の場合、除外値の指定は参照されません。

"True"の場合、次項以降で説明する<except>ノードまたは<exceptRange>ノードの記述に従って除外値を設定します (最後に記述された<except>ノードまたは<exceptRange>ノードが有効になります)。

(3-g) <except>ノード

<except>ノードは、<param>ノードのパラメータ値に対する除外値を直接記述するノードです。

[記述形式]

```
<except>除外値</except>
```

除外値は、空白で区切って複数指定することが可能です。尚、<except>ノードの記述は<param>ノードの type 属性が"int"または"real"の場合以外は無視されます。

(3-h) <exceptRange>ノード

<exceptRange>ノードは、<param>ノードのパラメータ値に対する除外値を範囲で指定するノードです。

[記述形式]

```
<except min="最小値" max="最大値" />
```

min 属性に除外範囲の最小値を、max 属性に最大値を記述します。min 属性または max 属性の記述を省略した場合、省略された方向の除外範囲制限は無しになります。

尚、<exceptRange>ノードの記述は<param>ノードの type 属性が"int"または"real"の場合以外は無視されます。

(3-i) <arithPrec>ノード

<arithPrec>ノードは、<param>ノードのパラメータを MOEA の設計変数として使用する場合に、変数の精度 (小数点以下の桁数) を指定するノードです。

[記述形式]

```
<arithPrec>桁数</arithPrec>
```

記述を省略した場合、変数の精度は<param>ノードの type 属性が"int"の場合は 0、"real"の場合は 6 になります。

尚、<arithPrec>ノードの記述は<param>ノードの type 属性が"int"または"real"の場合以外は無視されます。

(3-j) 依存関係記述ノード

<param>ノード配下には、他のパラメータ項目との依存関係を示すためのサブノードを記述することができます。依存関係記述ノードの記述方法については、(5) を参照して下さい。

(4) パラメータのグルーピング

複数のパラメータ項目をグループとしてまとめるため、<group>ノードを記述することができます。

[**記述形式**]

```
<group name="グループ名">
  { パラメータノード群 }
</group>
```

<group>ノードには、name 属性を記述する必要があります。name 属性には、グループ名を記述します。

<group>ノード配下には、グルーピングするパラメータ項目の<param>ノードを複数記述することができます。グルーピングされたパラメータ群は、PDI サブシステム上でパラメータ値の設定を行う際に、まとめて参照出来るように処理されます。

(5) 依存関係の記述

(5-a) <depend>ノード

<depend>ノードは、このノードを配下に持つ<param>ノード (親ノード) の記述が、他の<param>ノードのパラメータ値に依存することを示すためのノードです。

[**記述形式**]

```
<depend target="パラメータ項目名" [target2="パラメータ項目名 2"]>
  <cond>ノード
</depend>
```

<depend>ノードには、依存する<param>ノードの名前 (name 属性) を target および target2 属性に記述する必要があります。ここに記述した<param>ノードが存在しない場合、この<depend>ノードの記述は無視されます。target 属性は必ず記述する必要がありますが、target2 属性の記述は無くても構いません。

<depend>ノード配下には、依存する<param>ノードとの関係条件を記述する<cond>ノードをサブノードとして記述します。

尚、<depend>ノードは<param>ノードの配下にのみサブノードとして記述することができます。

(5-b) <cond>ノード

<cond>ノードは、パラメータ項目間の依存関係の条件を記述するためのノードで、<depend>ノードのサブノードとして記述します。

[記述形式]

<cond>「条件式」 ？ 「真文」 ： 「偽文」</cond>

ここで、「条件式」は以下の形式で記述します。

「ターゲット」「演算子」「値」
「ターゲット」「演算子」「値」「結合演算子」「ターゲット 2」「演算子 2」「値 2」

「ターゲット」には固定文字列として"VAL"を記述します。

「演算子」には、以下のいずれかを記述します。

「==」または「EQ」

「!=」または「NE」

「<」または「LT」

「<=」または「LE」

「>」または「GT」

「>=」または「GE」

(一部の Python/XML 処理系では、XML タグ以外で「<」文字を使用出来ません。)

<cond>ノードの上位の<depend>ノードで target2 属性が指定されている場合、ここで指定された<param>ノードの値を「値 2」として、2 つの条件式を「結合演算子」で結合する事が出来ます。

「結合演算子」以下のいずれかを記述します。

「&&」または「AND」

「||」または「OR」

ターゲットに指定されたのパラメータの type 属性が"string"または"choice"の場合、記述できる演算子は「==」、「!=」、「<」および「>」のみで、「==」が文字列の一致を、それ以外は文字列の不一致を表します。

「真文」および「偽文」には、以下のいずれかを記述することができます。

「enable」

「disable」

「set value="値"」

「set range="最小値 最大値"」

「set sweep="最小値 最大値 刻み幅"」 ここで、「enable」および「disable」は、このパラメータ項目の有効化／無効化を表します。また、「set value」、「set range」および「set sweep」は、このパラメータ項目の値、値域およびパラメータスイープ範囲と刻み幅を設定 (変更) することを意味します。

(6) パラメータサーベイ方式設定情報

パラメータサーベイ方式の設定情報は、<survey>ノードに記述されます。

[記述形式]

<survey>

{ サブノード群 }

</survey>

<survey>ノード配下に記述できるサブノードについては、以下の説明を参照して下さい。

(6-a) <wdPattern>ノード

<wdPattern>ノードは、サブケースジョブの作業ディレクトリ名のパターンを記述するノードです。

[記述形式]

<wdPattern>作業ディレクトリ名パターン</wdPattern>

(6-b) <pfPattern>ノード

<pfPattern>ノードは、サブケースジョブのソルバ入力パラメータファイル名のパターンを記述するノードです。

[記述形式]

<pfPattern>ソルバ入力パラメータファイル名パターン</pfPattern>

(6-c) <moeaMode>ノード

<moeaMode>ノードは、MOEA 計算を有効にするかどうかを記述するノードです。

[記述形式]

<moeaMode> True | False </moeaMode>

"True"が記述された場合、MOEA 計算を有効に設定します。

(6-d) <moea>ノード

<moea>ノードは、MOEA 計算を実行する際の各種設定値を記述するノードです。

[記述形式]

<moea>

{ サブノード群 }

</moea>

"<moea>"ノード配下には、MOEA 計算用の各種設定を記述するための各種ノード群をサブノードとして記述します。

(6-e) <population>ノード

<population>ノードは、MOEA 計算の個体数を記述するノードです。

[記述形式]

<population>個体数</population>

個体数は 1 以上の 4 の倍数の整数で指定します。MOEA 計算を行う場合、このノードの記述は省略できません。

(6-f) <maxGeneration>ノード

<maxGeneration>ノードは、MOEA 計算の最大世代数を記述するノードです。

[記述形式]

<maxGeneration>最大世代数</maxGeneration>

最大世代数は 1 以上の整数で指定します。

<maxGeneration>ノードの記述を省略した場合のデフォルト値は 1 です。

(6-g) <evaluator>ノード

<evaluator>ノードは、MOEA 計算における評価モジュール (evaluator プログラム) の実行パスを記述するノードです。

[記述形式]

<evaluator>evaluator プログラムの実行パス</evaluator>

evaluator プログラムの実行パスは、絶対パスか、ケースディレクトリからの相対パスで指定します。

(6-h) <optimizer>ノード

<optimizer>ノードは、MOEA 計算における最適化モジュール (optimizer プログラム) の実行パスを記述するノードです。

[記述形式]

<optimizer>optimizer プログラムの実行パス</optimizer>

optimizer プログラムの実行パスは、絶対パスか、ケースディレクトリからの相対パスで指定します。

<optimizer>ノードの記述を省略した場合のデフォルト値は、PDI 付属の Cheetah を使用するプログラム (xpdi_moea_cheetah) です。

(6-i) <randSeed>ノード

<randSeed>ノードは、MOEA 計算における乱数の種を記述するノードです。

[記述形式]

<randSeed>乱数の種</randSeed>

乱数の種は 0.0 〜 1.0 の実数で指定します。負値が指定された場合は自動で指定されます。

<randSeed>ノードの記述を省略した場合のデフォルト値は-1.0 です。

(6-j) <crossoverRate>ノード

<crossoverRate>ノードは、MOEA 計算における交叉率を記述するノードです。

[記述形式]

<crossoverRate>交叉率</crossoverRate>

交叉率は 0.0 〜 1.0 の実数で指定します。

<crossoverRate>ノードの記述を省略した場合のデフォルト値は 0.9 です。

(6-k) <mutationRate>ノード

<mutationRate>ノードは、MOEA 計算における突然変異率を記述するノードです。

[記述形式]

<mutationRate>突然変異率</mutationRate>

突然変異率は 0.0 〜 1.0 の実数で指定します。

<mutationRate>ノードの記述を省略した場合のデフォルト値は (1.0/設計変数個数) です。

(6-l) <etaCrossover>ノード

<etaCrossover>ノードは、MOEA 計算における交叉の distribution index を記述するノードです。

[記述形式]

<etaCrossover>distribution index</etaCrossover>

distribution index は 5 〜 40 の整数で指定します。

<etaCrossover>ノードの記述を省略した場合のデフォルト値は 25 です。

(6-m) <etaMutation>ノード

<etaMutation>ノードは、MOEA 計算における突然変異の distribution index を記述するノードです。

[記述形式]

<etaMutation>distribution index</etaMutation>

distribution index は 5 〜 50 の整数で指定します。

<etaMutation>ノードの記述を省略した場合のデフォルト値は 20 です。

(6-n) <history>ノード

<history>ノードは、MOEA 計算において全評価個体の履歴ファイル (interface/history.txt) を作成するかどうかを記述するノードです。

[記述形式]

```
<history> True | False </history>
```

True の場合、履歴ファイルを作成します。

<history>ノードの記述を省略した場合のデフォルト値は False です。

(6-o) <duplicate>ノード

<duplicate>ノードは、MOEA 計算において重複する設計点を作成するかどうかを記述するノードです。

[記述形式]

```
<duplicate> True | False </duplicate>
```

True の場合、重複する設計点を作成します。

<duplicate>ノードの記述を省略した場合のデフォルト値は False です。

(7) PDI 設定情報

上記以外の PDI の設定情報は、<snapshot>ノードに記述されます。

[記述形式]

```
<snapshot>
  { サブノード群 }
</snapshot>
```

<snapshot>ノード配下に記述できるサブノードについては、以下の説明を参照して下さい。

(7-a) <desc_path>ノード

<desc_path>ノードは、PDI にロードされたパラメータ定義ファイルのパスを記述するノードです。

[記述形式]

```
<desc_path>パラメータ定義ファイルのパス</desc_path>
```

パラメータ定義ファイルのパスは絶対パスか、ケースディレクトリからの相対パスで指定します。

尚、<snapshot>ノード配下に複数の<desc_path>ノードが記述された場合、最後に記述された <desc_path>ノードが有効になります。

(7-b) <templ_path>ノード

<templ_path>ノードは、PDI に登録されたテンプレートファイルのパスを記述するノードです。

[**記述形式**]

```
<templ_path>テンプレートファイルのパス</templ_path>
```

テンプレートファイルのパスは絶対パスか、ケースディレクトリからの相対パスで指定します。
<templ_path>ノードは<snapshot>ノード配下に複数記述することができます。

4.2 ソルバ入力パラメータテンプレートファイル

ソルバ入力パラメータテンプレートファイルは、ソルバ入力パラメータファイルの雛形であり、PDI で設定されたパラメータ値を埋め込むためのマクロ記述を含むものです。

以下に、テンプレートファイル中のマクロ記述形式を示します。

(一次変換式)

%パラメータ名 [！デフォルト値] [＊数値 1] [＋数値 2]%

(Python 式)

%パラメータ名 [！デフォルト値] [：式 (VAL)]%

PDI による入力パラメータファイル生成時に、「%パラメータ名%」は PDI 上で設定されたパラメータ値に置き換えられます。

ここで、「パラメータ名」はパラメータ定義ファイルの<param>ノードの名前 (name 属性値) で、デフォルト値はパラメータ定義ファイル中に name 属性が「パラメータ名」である<param>ノードが存在しない場合のパラメータ値です。デフォルト値は (「!」を含めて) 記述を省略することができます。

(一次変換式)

パラメータ項目のタイプが"int"または"real"の場合には、マクロ中に「演算子」と数値を記述することができます。これは、PDI 上で設定されたパラメータ値に対して一次変換を適用した結果を入力パラメータファイルに記述するためのもので、省略が可能です。入力パラメータファイルに記述される最終的な値は、以下のようになります。

(PDI で設定されたパラメータ値) × (数値 1) ＋ (数値 2)

パラメータ項目のタイプが"bool"の場合にも、マクロ中に「演算子」と数値を記述することができます。この場合、「数値 1」部分はパラメータ値が True の場合の記述文字列、「数値 2」部分はパラメータ値が False の場合の記述文字列と解釈され、省略時にはそれぞれ"True", "False"になります。

(Python 式)

マクロ記述中に「：」が含まれる場合、「：」以降の部分について「VAL」を PDI 上で設定されたパラメータ値とする Python 形式の計算式として評価し、マクロ記述と置き換えます。

[マクロ展開例]

マクロ記述

```
Nu = %Reynolds !  10 :  0.1 * 1.0 / VAL%
```

PDI 上での設定値

パラメータ名=”Reynolds”、値=100.0

Python 式の値

$0.1 \times 1.0/100.0 = 0.001$

最終的な記述

```
Nu = 0.001
```

4.3 パラメータリストファイル

パラメータリストファイルは、現在実行されているパラメータスイープのパラメータ値およびスコアファイルのパスを記述したファイルです。以下に示す情報が格納された CSV 形式のテキストファイルで、デフォルトでは「ケースディレクトリ/param_list.csv」に置かれます。

1 行目 現在の世代番号，最大世代数

2 行目 パラメータ項目数，パラメータ名 1，パラメータ名 2，...，パラメータ名 N

3 行目 スコアファイル名，パラメータ値 1，パラメータ値 2，...，パラメータ値 N

...

パラメータリストファイルの 1 行目には、現在の世代番号 (第 1 世代は 0) と最大世代数が格納されます。最大世代数が記述されていない場合は、世代数の制限なしを意味します。

2 行目には、パラメータ項目数 (=N) と、スイープするパラメータ名 (N 個) が記述されます。

3 行目以降には、各サブケース毎のスコアファイル名 (サブケースディレクトリ名/スコアファイル名) と、そのサブケースにおけるパラメータ値 (N 個) が記述されます。

5 外部プログラムの形式

5.1 evaluator プログラム

Cheetah による MOEA 計算は、制約条件値が負値にならない条件下で目的関数値を最小化する最適化問題を扱います。

evaluator は、MOEA 計算において各サブケースの解析結果から目的関数および制約条件の値を計算し、ファイルに出力するプログラムで、MOEA 計算実行スクリプトから呼び出されます。

HPC/PF システムでは evaluator プログラムのインターフェースのみを定義します。使用者は下記の仕様に則って、evaluator プログラムを最適化問題毎に実装する必要があります。

コマンドライン形式

evaluator `[-x case_dir] [-w wkd_base]`
`[-t tm_start] [-T tm_end] [-i interface_dir] [-p population]`
`[-O | -C]`

引数説明

`-x case_dir` ケースディレクトリ指定
ケースディレクトリを指定します。指定が省略された場合は、カレントワーキングディレクトリがケースディレクトリとなります。

`-w wkd_base` サブケースディレクトリのベース部分の指定
各サブケースディレクトリのベース部分 (共通部分) を指定します。サブケースディレクトリは、「ケースディレクトリ/ベース部分_数字」の形式で存在するものと想定されます。省略時は、「`job`」が指定されたものとみなされます。

`-t tm_start` 評価対象の開始時刻指定

`-T tm_end` 評価対象の終了時刻指定
evaluator がサブケース毎の解析結果を評価する際に、評価の対象とする時刻範囲を実数値で指定します。省略時は、開始時刻は 0.0、終了時刻は最終時刻となります。

`-i interface_dir` MOEA の作業用ディレクトリ指定
evaluator と optimizer がデータを受け渡すための作業用ディレクトリを、ケースディレクトリからの相対パス、または絶対パスで指定します。省略時は「`interface`」となります。
evaluator プログラムは、目的関数の評価結果を `interface_dir/pop_objs_eval.txt` というテキストファイルに、1 行 1 個体の目的関数値を空白またはタブ区切りで出力する必要があります。また、evaluator プログラムは、制約条件の評価結果を `interface_dir/pop_cons_eval.txt` というテキストファイルに、1 行 1 個体の制約条件値を空白またはタブ区切りで出力する必要があります。

`-p population` MOEA の個体数指定
MOEA 計算における個体数を 1 以上の 4 の倍数の整数で指定します。省略時は、「`-w`」で指定したサブケースディレクトリのベース部分を含むディレクトリの数となります。

尚、このオプションで指定された個体数と、ケースディレクトリ配下のサブケースディレクトリのベース部分を含むディレクトリの数が一致しない場合はエラーとなります。

`-O` 目的関数の数を返す
この evaluator が評価する目的関数の数 (1 以上の整数値) を標準出力に出力して終了します。

`-C` 制約条件の数を返す
この evaluator が評価する制約条件の数 (整数値) を標準出力に出力して終了します。制約条件が無い場合は 0 を出力します。

戻り値 (終了ステータス)

0 正常終了

0 以外 異常終了

尚、PDI にはサンプルの evaluator プログラムとして `ffvc_eval_2Dcyl` が付属しています。このプログラムは、FFV-C の 2 次元角柱周り流れの組込み問題の第 2 角柱が受ける外力を評価する Python プログラムで、Python コードは `pdi/lib/python/ffvc_eval_2Dcyl.py` です。

5.2 optimizer プログラム

optimizer は、MOEA 計算において目的関数および制約条件の値から、次の世代の設計変数を進化計算によって求めるプログラムで、MOEA 計算実行スクリプトから呼び出されます。

HPC/PF システムでは optimizer プログラムのインターフェースを定義しています。使用者は、下記の仕様に則って optimizer プログラムを実装することができます。

コマンドライン形式

optimizer [-x case_dir] [-i interface_dir]
-p population -c cur_gen -n max_gen [-m moea_comm]
[--cfg-moea=moea_cfg] [--cfg-mop=mop_cfg] [-s rand_seed]
[--nohistory] [--noduplicate]

引数説明

-x case_dir ケースディレクトリ指定
ケースディレクトリを指定します。指定が省略された場合は、カレントワーキングディレクトリがケースディレクトリとなります。

-i interface_dir MOEA の作業用ディレクトリ指定
evaluator と optimizer がデータを受け渡すための作業用ディレクトリを、ケースディレクトリからの相対パス、または絶対パスで指定します。省略時は「interface」となります。
optimizer プログラムは、目的関数および制約条件の評価結果を interface_dir/[pop_objs_eval.txt, pop_cons_eval.txt] というテキストファイルから入力し、生成した次世代の設計変数を interface_dir/pop_vars_eval.txt というテキストファイルに、1 行 1 個体の設計変数値を空白またはタブ区切りで出力する必要があります。

-p population MOEA の個体数指定
MOEA 計算における個体数を 1 以上の 4 の倍数の整数で指定します。この指定は省略できません。

-c cur_gen MOEA 計算の世代番号
MOEA 計算における現在の世代番号を 0 以上の整数で指定します。この指定は省略できません。

-n max_gen MOEA 計算の最大世代番号
MOEA 計算における最大世代番号を 1 以上の整数で指定します。この指定は省略できません。

-m moea_comm 進化計算プログラムのパス
optimizer プログラムが内部的に使用する進化計算プログラムのパスを指定します。省略時にはデフォルトの進化計算プログラムを使用します。

--cfg-moea=moea_cfg 進化計算設定ファイルのパス
optimizer プログラムが内部的に使用する進化計算プログラムの設定ファイルのパスを

指定します。省略時には cfg/moea.cfg となります。

--cfg-mop=mop_cfg 最適化問題設定ファイルのパス
optimizer プログラムが内部的に使用する進化計算プログラムが参照する最適化問題設定ファイルのパスを指定します。省略時には cfg/mop.cfg となります。

-n rand_seed 乱数の種
乱数の種を 0.0 〜 1.0 の実数で指定します。省略時には自動的に設定されます。

--nohistory history ファイルを作らない
このオプションを指定すると、全評価個体の履歴ファイル (interface/history.txt) を作成しません。

--noduplicate 重複する設計点を作らない
このオプションを指定すると、重複する設計点を作成しません。

戻り値 (終了ステータス)

0 正常終了

0 以外 異常終了

PDI システムでは、上記の仕様に従い、内部的に Cheetah を使用する optimizer プログラムとして xpdi_moea_cheetah を提供しています。

xpdi_moea_cheetah は、PDI と同じディレクトリに存在しています。xpdi_moea_cheetah では、「-m」オプション指定を省略すると Cheetah を進化計算プログラムとして使用するように実装されています。

5.3 xpdi_genparam プログラム

xpdi_genparam は、設計変数記述テキストファイル (interface_dir/pop_vars_eval.txt) を入力し、パラメータ記述 XML ファイルとソルバー入力テンプレートファイルを参照して、各サブケースディレクトリ配下にソルバー入力パラメータファイルを生成するプログラムで、MOEA 計算実行スクリプトファイル中から内部的に呼び出されます。

以下に xpdi_genparam プログラムの仕様を示します。

コマンドライン形式

```
xpdi_genparam  [-x case_dir] [-i interface_dir] [-f paramfile_pat]
  -d desc_file  [-t tmpl_file]+ [-p param_name]+
```

引数説明

`-x case_dir` ケースディレクトリ指定
: ケースディレクトリを指定します。指定が省略された場合は、カレントワーキングディレクトリがケースディレクトリとなります。

`-i interface_dir` MOEA の作業用ディレクトリ指定
: pop_vars_eval.txt ファイルが存在するディレクトリを、ケースディレクトリからの相対パス、または絶対パスで指定します。省略時は「interface」となります。

`-f paramfile_pat` ソルバー入力パラメータファイル名のパターン指定
: 各サブケースディレクトリ配下に生成される、ソルバー入力パラメータファイルのファイル名パターンを指定します。省略時は「%T」となり、ソルバー入力テンプレートファイル名から末尾の".template", ".tmpl", ".tmp", ".tpl"を除いたものになります。ファイル名のパターン指定の詳細については、「2.3 パラメータサーベイ方式設定機能」の「parameter file pattern」についての説明を参照して下さい。

`-d desc_file` パラメータ記述 XML ファイル指定
: パラメータ記述 XML ファイルのパスを指定します。絶対パスか、ケースディレクトリからの相対パスで指定します。この指定は省略できません。

`-t template_file` ソルバ入力パラメータテンプレートファイル指定
: パラメータテンプレートファイルのパスを指定します。絶対パスか、ケースディレクトリからの相対パスで指定します。複数指定が可能です。少なくとも 1 個以上の指定が必要です。

`-p param_name` パラメータ名の指定
: 設計変数として使用するパラメータ名指定します。複数指定が可能です。少なくとも 1 個以上の指定が必要です。

戻り値 (終了ステータス)

0　　正常終了

0 以外　異常終了

xpdi_genparam は PDI と同じディレクトリに存在しています。

尚、使用者が直接 xpdi_genparam プログラムを実行する必要も、また PDI 上で xpdi_genparam のパスを設定する必要もありません。